Some NP-Complete Problems

Six basic genres of NP-complete problems and paradigmatic examples.
- Packing problems: SET-PACKING, INDEPENDENT SET.
- Covering problems: SET-COVER, VERTEX-COVER.
- Constraint satisfaction problems: SAT, 3-SAT.
- Sequencing problems: HAMILTONIAN-CYCLE, TSP.
- Partitioning problems: 3D-MATCHING 3-COLOR.
- Numerical problems: SUBSET-SUM, KNAPSACK.

Practice. Most NP problems are either known to be in P or NP-complete.

Notable exceptions. Factoring, graph isomorphism, Nash equilibrium.
Polynomial-Time Reductions

3-SAT reduces to:
- INDEPENDENT SET
- GRAPH 3-COLOR
- SUBSET-SUM

3-SAT reduces to:
- VERTEX COVER
- HAM-CYCLE
- PLANAR 3-COLOR

Dick Karp (1972)
1985 Turing Award
INDEPENDENT SET: Given a graph $G = (V, E)$ and an integer $k$, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in $S$?

Ex. Is there an independent set of size $\geq 6$? Yes.
Ex. Is there an independent set of size $\geq 7$? No.
3 Satisfiability Reduces to Independent Set

Claim. \(3\text{-SAT} \leq_p \text{INDEPENDENT-SET}.\)

Pf. Given an instance \(\Phi\) of 3-SAT, we construct an instance \((G, k)\) of INDEPENDENT-SET that has an independent set of size \(k\) iff \(\Phi\) is satisfiable.

Construction.
- \(G\) contains 3 vertices for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.

\[
\Phi = \left( \overline{x_1} \lor x_2 \lor x_3 \right) \land \left( x_1 \lor \overline{x_2} \lor x_3 \right) \land \left( \overline{x_1} \lor x_2 \lor x_4 \right)
\]
3 Satisfiability Reduces to Independent Set

Claim. $G$ contains independent set of size $k = |\Phi|$ iff $\Phi$ is satisfiable.

Pf. $\Rightarrow$ Let $S$ be independent set of size $k$.
- $S$ must contain exactly one vertex in each triangle.
- Set these literals to true. and any other variables in a consistent way
- Truth assignment is consistent and all clauses are satisfied.

Pf $\Leftarrow$ Given satisfying assignment, select one true literal from each triangle. This is an independent set of size $k$. □

$\Phi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_4)$
**VERTEX COVER:** Given a graph $G = (V, E)$ and an integer $k$, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in $S$?

*Ex.* Is there a vertex cover of size $\leq 4$? Yes.

*Ex.* Is there a vertex cover of size $\leq 3$? No.
Claim. \( \text{VERTEX-COVER} \equiv_p \text{INDEPENDENT-SET} \).

Pf. We show \( S \) is an independent set iff \( V - S \) is a vertex cover.
Vertex Cover and Independent Set

Claim. VERTEX-COVER \equiv_p INDEPENDENT-SET.

Pf. We show $S$ is an independent set iff $V - S$ is a vertex cover.

\[ \Rightarrow \]
- Let $S$ be any independent set.
- Consider an arbitrary edge $(u, v)$.
- $S$ independent $\Rightarrow u \notin S$ or $v \notin S$ $\Rightarrow u \in V - S$ or $v \in V - S$.
- Thus, $V - S$ covers $(u, v)$.

\[ \Leftarrow \]
- Let $V - S$ be any vertex cover.
- Consider two nodes $u \in S$ and $v \in S$.
- Observe that $(u, v) \notin E$ since $V - S$ is a vertex cover.
- Thus, no two nodes in $S$ are joined by an edge $\Rightarrow S$ independent set. \qed
SET COVER: Given a set $U$ of elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of $U$, and an integer $k$, does there exist a collection of $\leq k$ of these sets whose union is equal to $U$?

Sample application.
- $m$ available pieces of software.
- Set $U$ of $n$ capabilities that we would like our system to have.
- The $i$th piece of software provides the set $S_i \subseteq U$ of capabilities.
- Goal: achieve all $n$ capabilities using fewest pieces of software.

Ex:

$$U = \{ 1, 2, 3, 4, 5, 6, 7 \}$$

$$k = 2$$

$$S_1 = \{3, 7\} \quad S_4 = \{2, 4\}$$

$$S_2 = \{3, 4, 5, 6\} \quad S_5 = \{5\}$$

$$S_3 = \{1\} \quad S_6 = \{1, 2, 6, 7\}$$
Vertex Cover Reduces to Set Cover

Claim. \( \text{VERTEX-COVER} \leq_p \text{SET-COVER} \).

Pf. Given a \( \text{VERTEX-COVER} \) instance \( G = (V, E), k \), we construct a set cover instance whose size equals the size of the vertex cover instance.

Construction.

- Create \( \text{SET-COVER} \) instance:
  - \( k = k \), \( U = E \), \( S_v = \{e \in E : e \text{ incident to } v \} \)
- Set-cover of size \( \leq k \) iff vertex cover of size \( \leq k \). □

\[ U = \{1, 2, 3, 4, 5, 6, 7\} \]
\[ k = 2 \]
\[ S_a = \{3, 7\} \quad S_b = \{2, 4\} \]
\[ S_c = \{3, 4, 5, 6\} \quad S_d = \{5\} \]
\[ S_e = \{1\} \quad S_f = \{1, 2, 6, 7\} \]
Review

Basic reduction strategies.
- Simple equivalence: \textsc{Independent-Set} $\equiv_p$ \textsc{Vertex-Cover}.
- Special case to general case: \textsc{Vertex-Cover} $\leq_p$ \textsc{Set-Cover}.
- Encoding with gadgets: \textsc{3-Sat} $\leq_p$ \textsc{Independent-Set}.

Transitivity. If $X \leq_p Y$ and $Y \leq_p Z$, then $X \leq_p Z$.

Pf idea. Compose the two algorithms.

Ex: \textsc{3-Sat} $\leq_p$ \textsc{Independent-Set} $\leq_p$ \textsc{Vertex-Cover} $\leq_p$ \textsc{Set-Cover}.
Self-Reducibility

Decision problem. Does there exist a vertex cover of size $\leq k$?
Search problem. Find vertex cover of minimum cardinality.

Self-reducibility. Search problem $\leq_P$ decision version.
- Applies to all (NP-complete) problems in this chapter.
- Justifies our focus on decision problems.

Ex: to find min cardinality vertex cover.
- (Binary) search for cardinality $k^*$ of min vertex cover.
- Find a vertex $v$ such that $G - \{v\}$ has a vertex cover of size $\leq k^* - 1$.
  - any vertex in any min vertex cover will have this property
- Include $v$ in the vertex cover.
- Recursively find a min vertex cover in $G - \{v\}$.

$\text{delete } v \text{ and all incident edges}$
**HAM-CYCLE**: given an undirected graph $G = (V, E)$, does there exist a simple cycle $\Gamma$ that contains every node in $V$.

**YES**: vertices and faces of a dodecahedron.
Hamiltonian Cycle

**HAM-CYCLE:** given an undirected graph $G = (V, E)$, does there exist a simple cycle $\Gamma$ that contains every node in $V$.

NO: bipartite graph with odd number of nodes.
**Directed Hamiltonian Cycle**

**DIR-HAM-CYCLE:** given a digraph $G = (V, E)$, does there exist a simple directed cycle $\Gamma$ that contains every node in $V$?

**Claim.** $\text{DIR-HAM-CYCLE} \leq_p \text{HAM-CYCLE}$.

**Pf.** Given a directed graph $G = (V, E)$, construct an undirected graph $G'$ with $3n$ nodes.

![Diagram of graphs $G$ and $G'$]
Claim. $G$ has a Hamiltonian cycle iff $G'$ does.

Pf. $\Rightarrow$
- Suppose $G$ has a directed Hamiltonian cycle $\Gamma$.
- Then $G'$ has an undirected Hamiltonian cycle (same order).

Pf. $\Leftarrow$
- Suppose $G'$ has an undirected Hamiltonian cycle $\Gamma'$.
- $\Gamma'$ must visit nodes in $G'$ using one of following two orders:
  - ..., $B$, $G$, $R$, $B$, $G$, $R$, $B$, $G$, $R$, $B$, ...
- Blue nodes in $\Gamma'$ make up directed Hamiltonian cycle $\Gamma$ in $G$, or reverse of one.
Claim. 3-SAT $\leq_p$ DIR-HAM-CYCLE.

Pf. Given an instance $\Phi$ of 3-SAT, we construct an instance of DIR-HAM-CYCLE that has a Hamiltonian cycle iff $\Phi$ is satisfiable.

Construction. First, create graph that has $2^n$ Hamiltonian cycles which correspond in a natural way to $2^n$ possible truth assignments.
3-SAT Reduces to Directed Hamiltonian Cycle

Construction. Given 3-SAT instance $\Phi$ with $n$ variables $x_i$ and $k$ clauses.
- Construct $G$ to have $2^n$ Hamiltonian cycles.
- Intuition: traverse path $i$ from left to right $\iff$ set variable $x_i = 1$. 

![Graph Diagram](image-url)
3-SAT Reduces to Directed Hamiltonian Cycle

Construction. Given 3-SAT instance $\Phi$ with $n$ variables $x_i$ and $k$ clauses.
- For each clause: add a node and 6 edges.

$C_1 = x_1 \lor \overline{x_2} \lor x_3$

$C_2 = \overline{x_1} \lor x_2 \lor \overline{x_3}$
3-SAT Reduces to Directed Hamiltonian Cycle

**Claim.** \( \Phi \) is satisfiable iff \( G \) has a Hamiltonian cycle.

**Pf. \( \Rightarrow \)**

- Suppose 3-SAT instance has satisfying assignment \( x^* \).
- Then, define Hamiltonian cycle in \( G \) as follows:
  - if \( x^*_i = 1 \), traverse row \( i \) from left to right
  - if \( x^*_i = 0 \), traverse row \( i \) from right to left
  - for each clause \( C_j \), there will be at least one row \( i \) in which we are going in "correct" direction to splice node \( C_j \) into tour
3-SAT Reduces to Directed Hamiltonian Cycle

Claim. \( \Phi \) is satisfiable iff \( G \) has a Hamiltonian cycle.

Pf. \( \Rightarrow \)

- Suppose \( G \) has a Hamiltonian cycle \( \Gamma \).
- If \( \Gamma \) enters clause node \( C_j \), it must depart on mate edge.
  - thus, nodes immediately before and after \( C_j \) are connected by an edge \( e \) in \( G \)
  - removing \( C_j \) from cycle, and replacing it with edge \( e \) yields Hamiltonian cycle on \( G - \{ C_j \} \)
- Continuing in this way, we are left with Hamiltonian cycle \( \Gamma' \) in \( G - \{ C_1, C_2, \ldots, C_k \} \).
- Set \( x^*_{i} = 1 \) iff \( \Gamma' \) traverses row \( i \) left to right.
- Since \( \Gamma \) visits each clause node \( C_j \), at least one of the paths is traversed in "correct" direction, and each clause is satisfied. \( \blacksquare \)
**Longest Path**

**SHORTEST-PATH.** Given a digraph $G = (V, E)$, does there exists a simple path of length at most $k$ edges?

**LONGEST-PATH.** Given a digraph $G = (V, E)$, does there exists a simple path of length at least $k$ edges?

**Claim.** $3$-SAT $\leq_P$ LONGEST-PATH.

**Pf 1.** Redo proof for DIR-HAM-CYCLE, ignoring back-edge from $t$ to $s$.

**Pf 2.** Show HAM-CYCLE $\leq_P$ LONGEST-PATH.
3-Dimensional Matching

**3D-MATCHING.** Given $n$ instructors, $n$ courses, and $n$ times, and a list of the possible courses and times each instructor is willing to teach, is it possible to make an assignment so that all courses are taught at different times?

<table>
<thead>
<tr>
<th>Instructor</th>
<th>Course</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wayne</td>
<td>COS 423</td>
<td>MW 11-12:20</td>
</tr>
<tr>
<td>Wayne</td>
<td>COS 423</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Wayne</td>
<td>COS 226</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Wayne</td>
<td>COS 126</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Tardos</td>
<td>COS 523</td>
<td>TTh 3-4:20</td>
</tr>
<tr>
<td>Tardos</td>
<td>COS 423</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Tardos</td>
<td>COS 423</td>
<td>TTh 3-4:20</td>
</tr>
<tr>
<td>Kleinberg</td>
<td>COS 226</td>
<td>TTh 3-4:20</td>
</tr>
<tr>
<td>Kleinberg</td>
<td>COS 226</td>
<td>MW 11-12:20</td>
</tr>
<tr>
<td>Kleinberg</td>
<td>COS 423</td>
<td>MW 11-12:20</td>
</tr>
</tbody>
</table>
3-3imensional Matching

3D-MATCHING. Given disjoint sets X, Y, and Z, each of size n and a set $T \subseteq X \times Y \times Z$ of triples, does there exist a set of n triples in T such that each element of $X \cup Y \cup Z$ is in exactly one of these triples?

Claim. 3-SAT $\leq_P$ INDEPENDENT-COVER.
Pf. Given an instance $\Phi$ of 3-SAT, we construct an instance of 3D-matching that has a perfect matching iff $\Phi$ is satisfiable.
3-Dimensional Matching

**Construction.** (part 1)

- Create gadget for each variable $x_i$ with $2k$ core and tip elements.
- No other triples will use core elements.
- In gadget $i$, 3D-matching must use either both grey triples or both blue ones.

\[ x_1 \quad x_3 \quad x_2 \]

\[ \text{true} \quad \text{false} \quad \text{false} \]

number of clauses

\[ k = 2 \text{ clauses} \]
\[ n = 3 \text{ variables} \]

set $x_i = \text{true}$
set $x_i = \text{false}$
3-Dimensional Matching

**Construction.** (part 2)

- For each clause $C_j$ create two elements and three triples.
- Exactly one of these triples will be used in any 3D-matching.
- Ensures any 3D-matching uses either (i) grey core of $x_1$ or (ii) blue core of $x_2$ or (iii) grey core of $x_3$.

\[ C_j = x_1 \lor \overline{x_2} \lor x_3 \]
**Construction.** (part 3)
- For each tip, add a cleanup gadget.
**3-Dimensional Matching**

**Claim.** Instance has a 3D-matching iff $\Phi$ is satisfiable.

**Detail.** What are $X$, $Y$, and $Z$? Does each triple contain one element from each of $X$, $Y$, $Z$?
Claim. Instance has a 3D-matching iff $\Phi$ is satisfiable.

Detail. What are $X$, $Y$, and $Z$? Does each triple contain one element from each of $X$, $Y$, $Z$?
3-Colorability

3-COLOR: Given an undirected graph $G$ does there exists a way to color the nodes red, green, and blue so that no adjacent nodes have the same color?
Register Allocation

Register allocation. Assign program variables to machine register so that no more than k registers are used and no two program variables that are needed at the same time are assigned to the same register.

Interference graph. Nodes are program variables names, edge between u and v if there exists an operation where both u and v are "live" at the same time.

Observation. [Chaitin 1982] Can solve register allocation problem iff interference graph is k-colorable.

Fact. $3$-COLOR $\leq_p$ k-REGISTER-ALLOCATION for any constant $k \geq 3$. 
3-Colorability

Claim. 3-SAT \leq_p 3-COLOR.

Pf. Given 3-SAT instance \( \Phi \), we construct an instance of 3-COLOR that is 3-colorable iff \( \Phi \) is satisfiable.

Construction.

i. For each literal, create a node.
ii. Create 3 new nodes T, F, B; connect them in a triangle, and connect each literal to B.
iii. Connect each literal to its negation.
iv. For each clause, add gadget of 6 nodes and 13 edges.

↑

to be described next
3-Colorability

**Claim.** Graph is 3-colorable iff \( \Phi \) is satisfiable.

**Pf.**  \( \implies \) Suppose graph is 3-colorable.
- Consider assignment that sets all \( T \) literals to true.
- (ii) ensures each literal is \( T \) or \( F \).
- (iii) ensures a literal and its negation are opposites.

![Diagram of 3-Colorability](image)
3-Colorability

Claim. Graph is 3-colorable iff $\Phi$ is satisfiable.

Pf. $\Rightarrow$ Suppose graph is 3-colorable.

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.
- (iv) ensures at least one literal in each clause is T.

$$C_i = x_1 \lor \overline{x_2} \lor x_3$$
Claim. Graph is 3-colorable iff $\Phi$ is satisfiable.

Pf. $\Rightarrow$ Suppose graph is 3-colorable.
   
   - Consider assignment that sets all $T$ literals to true.
   - (ii) ensures each literal is $T$ or $F$.
   - (iii) ensures a literal and its negation are opposites.
   - (iv) ensures at least one literal in each clause is $T$.

\[ C_i = x_1 \lor \overline{x_2} \lor x_3 \]
3-Colorability

Claim. Graph is 3-colorable iff $\Phi$ is satisfiable.

Pf. $\Leftarrow$ Suppose 3-SAT formula $\Phi$ is satisfiable.
- Color all true literals $T$.
- Color node below green node $F$, and node below that $B$.
- Color remaining middle row nodes $B$.
- Color remaining bottom nodes $T$ or $F$ as forced. □

A literal set to true in 3-SAT assignment

$$C_i = x_1 \lor \overline{x_2} \lor x_3$$
**SUBSET-SUM.** Given natural numbers \( w_1, \ldots, w_n \) and an integer \( W \), is there a subset that adds up to exactly \( W \)?

**Ex:** \( \{ 1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344 \} \), \( W = 3754 \).

**Yes.** \( 1 + 16 + 64 + 256 + 1040 + 1093 + 1284 = 3754 \).

**Remark.** With arithmetic problems, input integers are encoded in binary. Polynomial reduction must be polynomial in binary encoding.

**Claim.** \( 3\text{-SAT} \leq_p \text{SUBSET-SUM} \).

**Pf.** Given an instance \( \Phi \) of 3-SAT, we construct an instance of SUBSET-SUM that has solution iff \( \Phi \) is satisfiable.
**Subset Sum**

**Construction.** Given 3-SAT instance Φ with n variables and k clauses, form $2n + 2k$ decimal integers, each of $n+k$ digits, as illustrated below.

**Claim.** Φ is satisfiable iff there exists a subset that sums to W.

**Pf.** No carries possible.

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>C₁</th>
<th>C₂</th>
<th>C₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>¬x</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>¬y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>z</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>¬z</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

|      | 0  | 0  | 1  | 0  | 0  | 0  | 100     |
|      | 0  | 0  | 2  | 0  | 0  | 0  | 200     |
|      | 0  | 0  | 0  | 1  | 0  | 0  | 10      |
|      | 0  | 0  | 0  | 2  | 0  | 0  | 20      |
|      | 0  | 0  | 0  | 0  | 1  | 0  | 1       |
|      | 0  | 0  | 0  | 0  | 2  | 0  | 2       |
| **W**| 1  | 1  | 1  | 4  | 4  | 4  | 111,444 |

$dummies to get clause columns to sum to 4$